IES de Teis
Avda. de Galicia, 101
36216 – Vigo

Tfno.: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

Sharding en MongoDB.

Llamamos sharding al método que distribuye los datos en múltiples nodos. Traducimos “shard” como fragmento, parte, pedazo y eso nos da una idea de lo que hace este método, dividir los datos en trozos que asignará a distintos servidores. Aquí es donde veremos claramente de una manera práctica que quiere decir eso de crecimiento horizontal.

Una base de datos alojada en un servidor no podrá crecer más que la capacidad de almacenamiento que tenga ese servidor. Mientras llega ese momento, la memoria y procesador tendrán que gestionar consultas con cada vez más datos por lo que puede que se acabarán saturando.

Uno de los métodos para gestionar este problema de crecimiento es el **escalado vertical** consistente en aumentar la capacidad del servidor con un disco de mayor capacidad, más memoria RAM o un mejor procesador. Todas estas opciones, aunque posibles, implican restricciones hardware que, en la práctica marcan un límite superior a la capacidad de ampliación vertical.

Otro de los métodos con los que gestionamos el crecimiento es el **escalado horizontal** consistente en dividir los datos en múltiples servidores. De esta manera, aunque la velocidad o capacidad de un único nodo no sea especialmente alta, este solo se estará encargando de una parte de los datos. Con un escaldo horizontal complicamos la infraestructura, pero permitimos expandir velocidad y capacidad sin más que añadir nodos a la red.

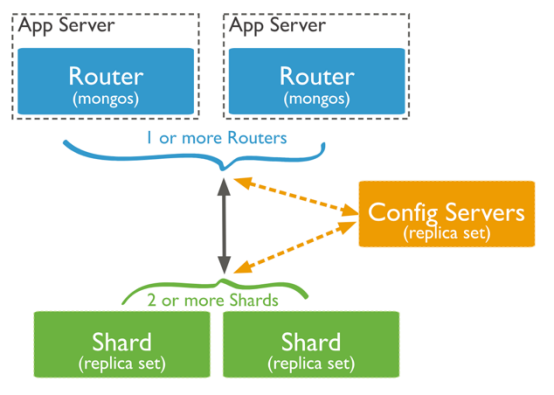
MongoDB proporciona escalado horizontal a través de sharding.

Como ventajas del sharding podemos decir que MongoDB distribuye las lecturas y escrituras sobre los shards del clúster. Aumenta la capacidad de almacenamiento con tan solo añadir nuevos shards al clúster. Usar réplicas en los config servers y el los shards proporciona alta disponibilidad.

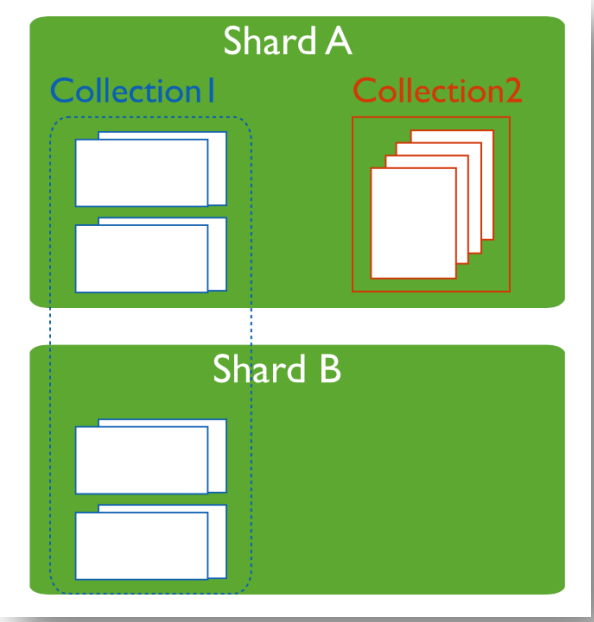
MongoDB proporciona mecanismos para volver a hacer un resharding cambiando las claves pero no proporciona ningún mecanismo para deshacer un sharding.

Un clúster de sharding tiene 3 componentes.

- Shards. Son los equipos que contienen los datos.
- Mongos. Son los nodos que actúan de interfaz entre las aplicaciones cliente y los shards. Funcionan como si fueran unos routers de consultas.
- Config Servers. Almacenan los metadatos y configuraciones del clúster.

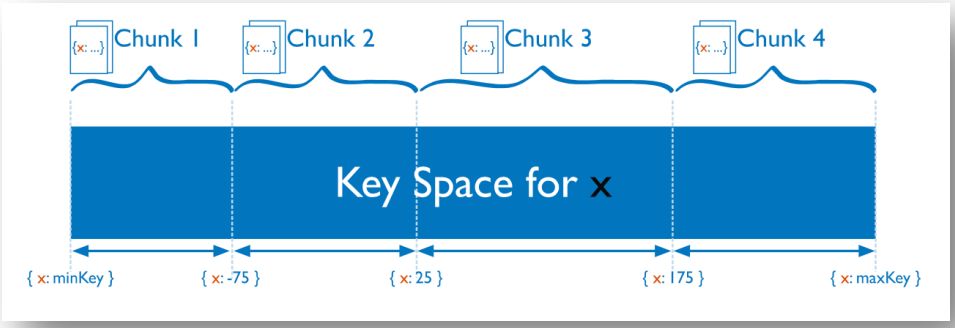


Los shards trabajan a nivel de colección distribuyendo los documentos de esa colección a través del clúster. Por esta misma razón es posible que una misma base de datos contenga algunas colecciones en shard y otras no. En estos casos las colecciones que no tienen shard se almacenarán en lo que se conoce como shard primario.



Shard Key

La llave shard es el o los campos que se utilizarán para distribuir los documentos de la colección por los distintos servidores del clúster shard. Estos campos deben tener un índice obligatoriamente.

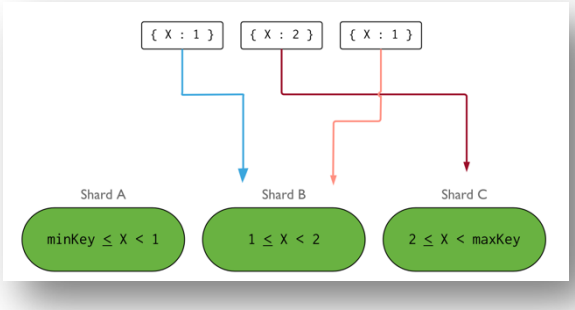


MongoDB dividirá los valores de la clave shard en rangos. Cada uno de estos rangos estará asociado a un "chunk" o fragmento.

El tamaño por defecto de un "chunk" es de 128 megabytes. Cuando los documentos que tendrían que ir es ese chunk ocupan más de ese tamaño, MongoDB los parte en dos nuevos fragmentos.

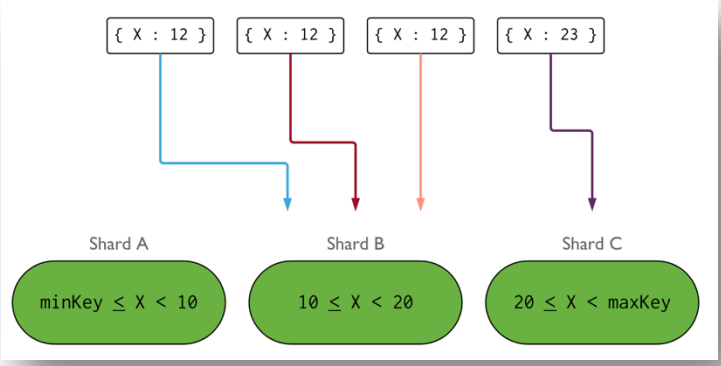
La elección de una clave adecuada es algo complicado porque afecta a la distribución de los chunks a través de todo el clústers. En el momento de elegir una clave de shard adecuada es necesario que tengamos en cuenta:

- **Cardinalidad**. Es el conjunto de elementos que la forman. Una cifra binaria tiene una cardinalidad de 2, una cifra decimal tiene una cardinalidad de 10. La cardinalidad del campo ciudades es mucho mayor que la cardinalidad del campo continentes. Para una clave shard buscaremos campos con cardinalidad alta ya que esto determina el límite máximo de chunks en el clúster. Si el modelo de datos que usamos tiene una clave con poca cardinalidad necesitaremos usar una llave compuesta con otros campos para aumentar la



cardinalidad y así poder dividir en más chunks.

- **Frecuencia**. Representa como de a menudo se repiten los mismos datos. Si la mayoría de los documentos están contenidos en un subconjunto pequeño de la clave de shard, el chunk que los contenga será un verdadero cuello de botella. Si ya no es posible subdividir el chunk porque no hay una clave con el que se pueda asociar el nuevo chunk, este seguirá creciendo, provocando grandes problemas de eficiencia en las consultas. A este chunk que crece sin poder dividirse se le conoce como jumbo chunk. En este escenario el escalado horizontal pierde su sentido porque, aunque añadamos nodos toda la carga la seguiría llevando el mismo.
- **Crecimiento monotónico**. Un clave con crecimiento monotónico siempre crecerá a un ritmo más o menos constante. Si elegimos clave una fecha de alta tendremos una clave con crecimiento monotónico. El resultado es que los nuevos documentos se irán guardando en el último shard hasta que llegue al límite de su tamaño y sea necesario partirlo en dos. Si no queremos este comportamiento podemos pensar en usar algún tipo de hash para conseguir mejor distribución.
- **Patrón de consultas**. Otro factor que hay que tener en cuenta a la hora de elegir una clave son las consultas que vamos a hacer. Si la llave aparece en la consulta, solo se enviará esa consulta a los shards que contienen



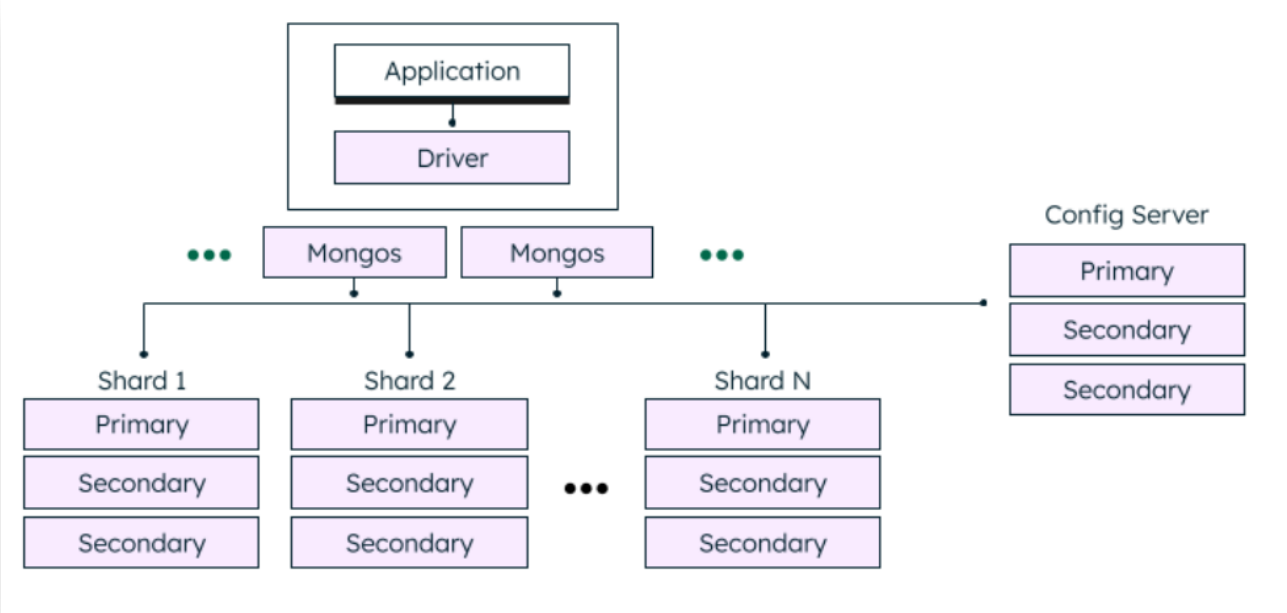
información relevante. En caso de que la llave no aparezca en la consulta obligará a que todos los shard ejecuten la consulta en su parte de los datos almacenados.

Para conectar con un clúster shard es obligatorio que los clientes se conecten a un router mongos. El proceso mongos es muy pequeño y actúa como interfaz entre las aplicaciones y el clúster shard además de manejar todo el balanceo y enrutado de consultas a través del clúster.

Conectar con mongos se hace de la misma manera que con mongod y podemos hacerlo a través de mongosh o cualquier otro cliente de mongoDB. Por ello, si quisiéramos averiguar si estamos conectados a un mongod o a un proceso mongos podemos ejecutar el comando “hello()” desde el shell. Sabremos que estamos conectados a un

proceso mongos, y por tanto a un clúster shard, si aparece en campo “msg” con el valor “isdbgrig”. Si no aparece este campo estaremos conectados a un mongod.

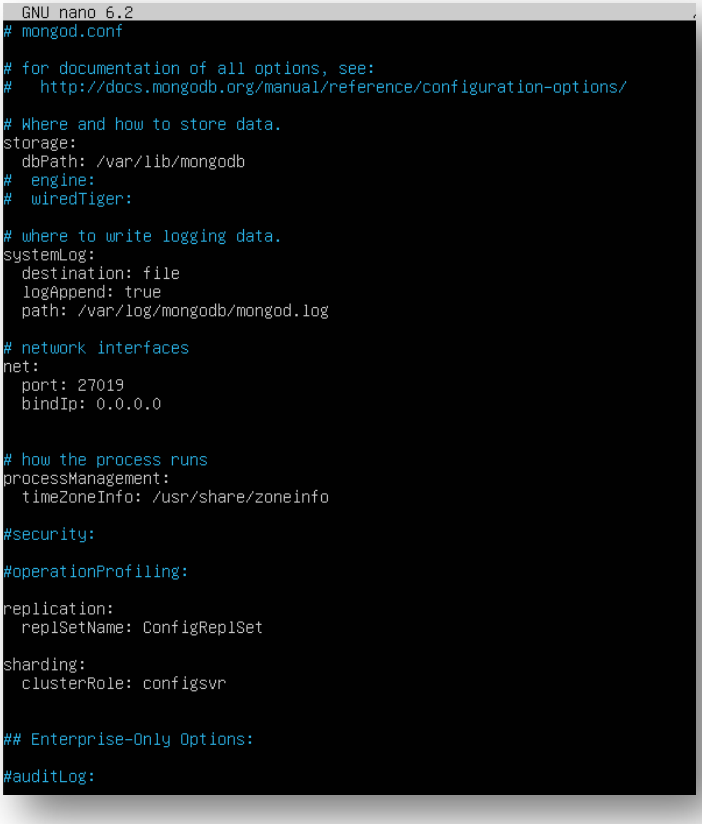
El escenario típico de un clúster shard es aquel en el que disponemos de uno o varios mongos. Los procesos mongos generan mucho tráfico con el config server así que no se recomienda más de 30 mongos. Para una mejor alta disponibilidad se recomienda que cada config server y cada shard sea a su vez un conjunto de réplica de 3 nodos.



El servidor de configuración se encarga de guardar los metadatos del clúster shard. Estos metadatos incluyen la lista de chunks en cada shard y los rangos que definen los chunks. Como este servidor se puede considerar un punto único de fallo se recomienda que se configuren como conjunto de réplicas. Hay una limitación a 3 servidores de configuración pero con un pequeño ajuste en la réplica podemos hackear esta limitación y tener hasta 50 servidores de configuración.

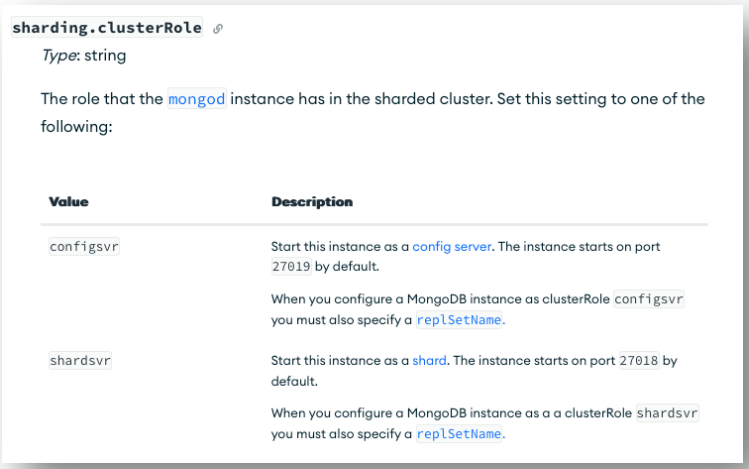
Partiremos de un escenario con 3 servidores MongoDB bien configurados con nombre e IPs fijas.

Configuraremos en el nodo1 el servidor de configuración, el nodo2 será el router de consultas y por último, el nodo3 será nuestro único shard.



El nodo1 funcionará como servidor de configuración. Este papel tenemos que declararlo en el archivo de configuración de MongoDB en "/etc/mongod.conf" dentro del apartado de sharding. Aquí indicaremos que la función que desempeñará este nodo será la de servidor de configuración mediante la línea "clusterRole: configsvr".

Los fragmentos y las réplicas van normalmente de la mano por lo que cuando declaramos un config server o un shard server tenemos que declarar también un nombre de réplica. En nuestro caso lo declararemos por imposición pero por simplificar no lo usaremos, tendremos un único primario en cada réplica.



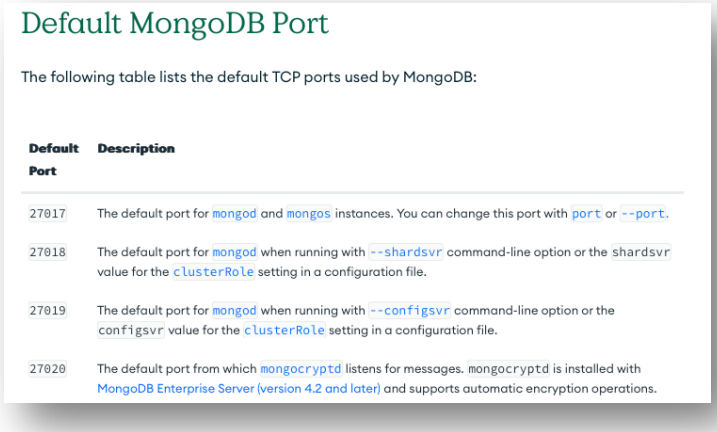
`sharding.clusterRole`

Type: string

The role that the `mongod` instance has in the sharded cluster. Set this setting to one of the following:

Value	Description
`configsvr`	Start this instance as a config server. The instance starts on port `27019` by default. When you configure a MongoDB instance as clusterRole `configsvr` you must also specify a `replSetName`.
`shardsvr`	Start this instance as a shard. The instance starts on port `27018` by default. When you configure a MongoDB instance as a a clusterRole `shardsvr` you must also specify a `replSetName`.

También modificaremos el puerto de escucha para adaptarnos a los puertos propuestos por MongoDB. En el caso de un config server, MongoDB propone usar el puerto 27019



Default MongoDB Port

The following table lists the default TCP ports used by MongoDB:

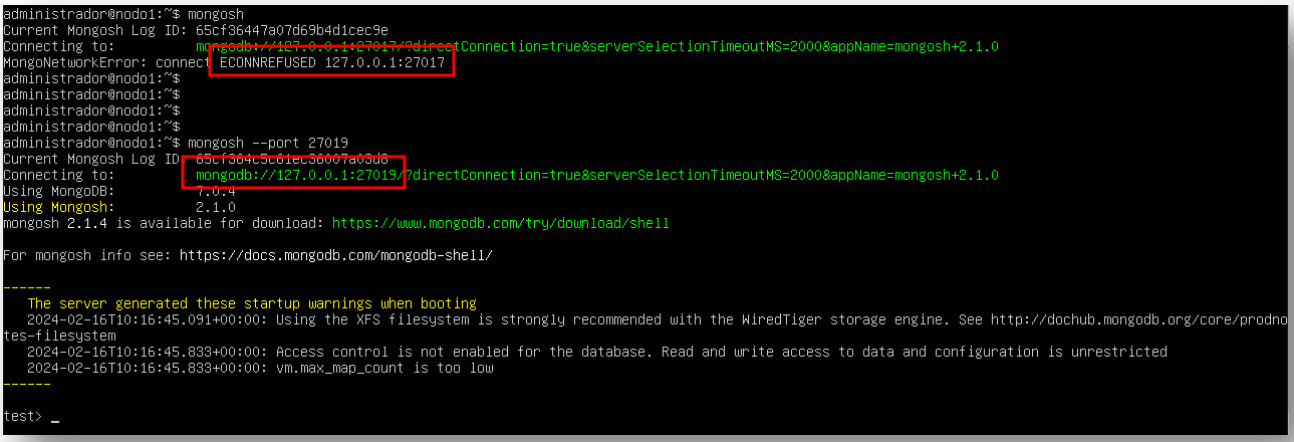
Default Port	Description
`27017`	The default port for `mongod` and `mongos` instances. You can change this port with `port` or `--port`.
`27018`	The default port for `mongod` when running with `--shardsvr` command-line option or the `shardsvr` value for the `clusterRole` setting in a configuration file.
`27019`	The default port for `mongod` when running with `--configsvr` command-line option or the `configsvr` value for the `clusterRole` setting in a configuration file.
`27020`	The default port from which `mongocryptd` listens for messages. `mongocryptd` is installed with MongoDB Enterprise Server (version 4.2 and later) and supports automatic encryption operations.

Para que los cambios tengan efecto será necesario reiniciar el nodo1 o reiniciar el servicio con el siguiente comando:

```
sudo systemctl restart mongod.service
```

Ahora que ya tenemos el archivo de configuración definido entramos al servidor usando mongosh. Recuerda que en este caso hemos cambiado el puerto original a través del cual nos conectábamos así que será necesario indicarlo en la llamada a mongosh.

```
mongosh –port 27019
```

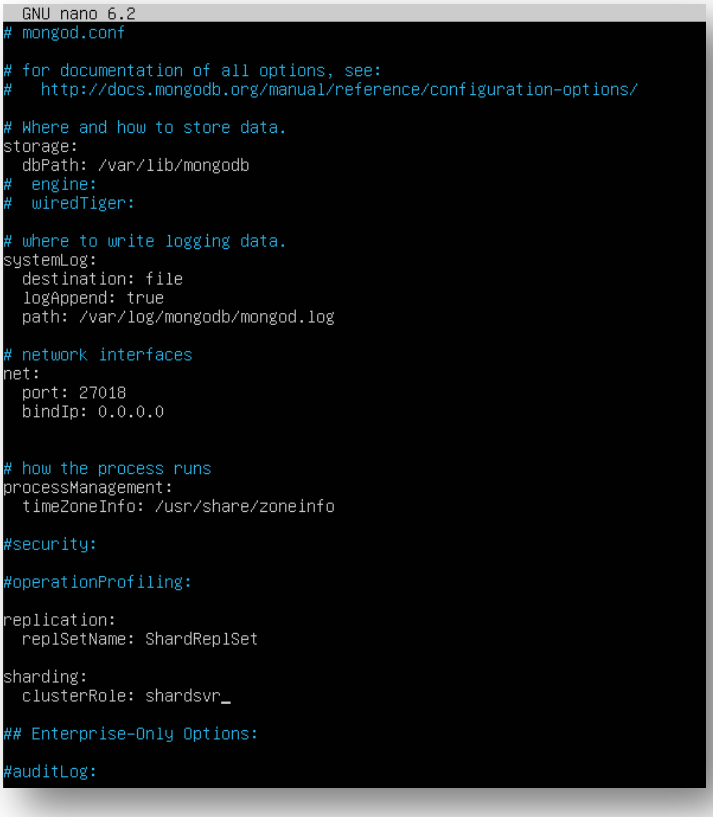


Una vez conectados iniciaremos el conjunto de réplica. En nuestro caso, como en este ejemplo vamos a hacer una configuración mínima, arrancaremos el conjunto de réplicas sin ningún dato. De esta manera estaremos configurando un único primario.

```
rs.initiate()
```

Como podemos ver en la captura anterior, al configurar un conjunto de réplicas vacío nos creará un único nodo primario al cabo un par de segundos.

Vamos a configurar ahora un shard en el nodo3. Al igual que con los config server, es necesario configurarlo como una réplica set. También es necesario especificar que el papel de desempeñará este nodo es el de servidor shard. Además, usaremos el puerto 27108 para las comunicaciones con el shard. Todo ello debe quedar indicado en el archivo de configuración como en la siguiente captura.



Tendremos que reiniciar el nodo3 o reiniciar el servicio mongod para que los cambios en el archivo de configuración tengan efecto.

Como lo hemos configurado como un conjunto de réplica, es necesario iniciarlo por primera vez para convertirlo en primario. Para ello nos conectamos al servidor de MongoDB teniendo en cuenta que en los servidores shard el puerto de escucha es el 27018.

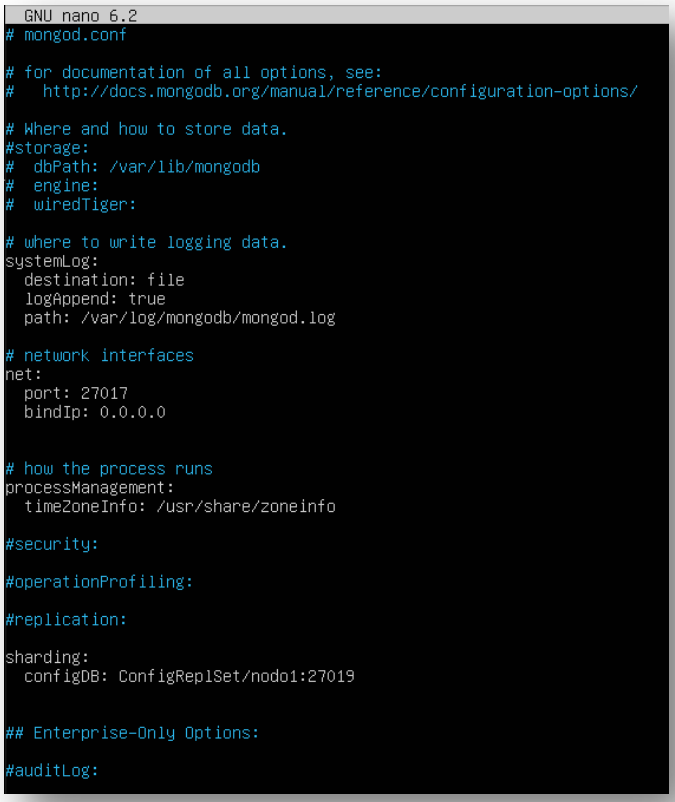
```
mongosh –port 27018
```

A continuación, una vez estemos dentro de la interfaz CLI del servidor MongoDB inicializamos la replicación con el siguiente comando:

```
rs.initiate()
```

Lo hacemos así, sin ningún argumento porque solo pretendemos tener un único primario.

Ahora vamos a configurar en nodo2 un router de consultas con mongos. Tendremos que modificar el archivo de configuración para indicarle donde está el servidor de configuración. Para ello incluimos en el apartado de sharding una línea con clave “configDB” y valor el nombre del conjunto de réplicas y la ip más puerto de alguno de sus nodos. Como nosotros solo tenemos un primario le daremos el nombre o dirección IP del nodo1 junto con el puerto 27019 que es el que usan por defecto los servidores de configuración en MongoDB.



Si nos fijamos en la captura anterior podemos ver como la parte de storage está comentada. Indicar un dbPath solo tiene sentido para mongod. Mongos no almacena bases de datos así que tendremos que eliminar estas opciones del archivo de configuración.

Este servidor está usando la ruta de logs por defecto. El problema es que estamos usando el usuario “administrador” y el propietario de la carpeta de logs es el usuario de sistema mongodb. Para cambiar el propietario de la carpeta de logs utilizaremos el siguiente comando:

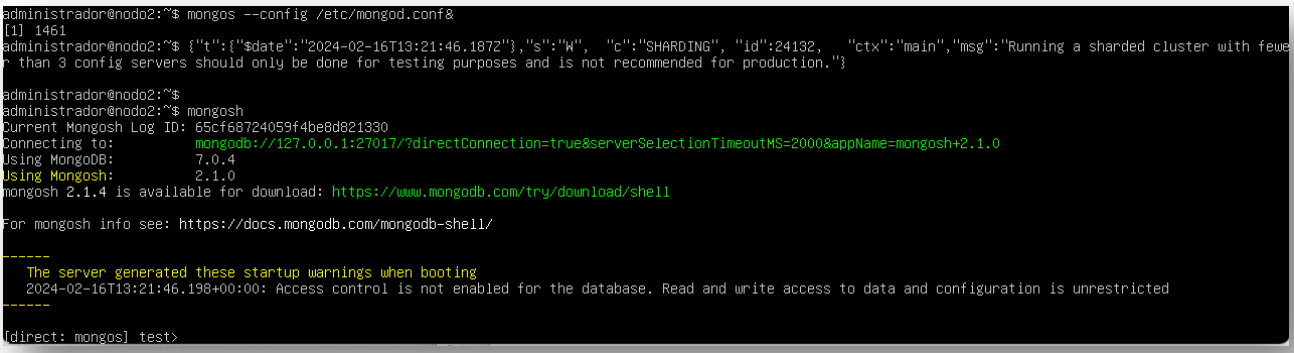
```
sudo chown -R administrador:administrador /var/log/mongodb
```

Vamos ahora a arrancar el servicio mongos haciendo referencia explícita al archivo de configuración que debe leer. Para no alargar la práctica no lo haremos pero aquí sería muy interesante configurar el arranque de mongos como un servicio para que se inicie solo en cada arranque del sistema.

Para arrancar mongos nos vamos al nodo2 y desde allí lo lanzamos en segundo plano.

```
mongos –config /etc/mongod.conf$
```

Una vez arrancado en segundo plano podremos conectarnos desde un mongosh desde la misma terminal



En principio ya tenemos los 3 nodos configurados con sus tres funciones.

- Nodo1 es el config server. Es un proceso mongod escuchando el puerto 27019. Internamente es un conjunto de réplica llamado “ConfigReplSet” con él mismo como único nodo y, por tanto, primario.
- Nodo2 es el query router. Es un proceso mongos escuchando en el puerto 27017.
- Nodo3 es un shard server. Es un proceso mongod escuchando en el puerto 27018. Internamente es un conjunto de réplica llamado “ShardReplSet”

El último paso consiste en añadir el servidor shard al clúster. Para ello nos vamos al servidor mongos y nos conectamos con un mongosh de la manera habitual.

```
sh.addShard(“ShardReplSet/nodo3:27018”)
```

```
[direct: mongos] test> sh.addShard("ShardReplSet/nodo3:27018")
{
  shardAdded: 'ShardReplSet',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1708095435, i: 6 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1708095435, i: 6 })
}
[direct: mongos] test>
```

Con este comando y la captura anterior añadimos al servidor de configuración, a través de mongos, un nuevo servidor shard. Para ello hemos indicado alguno de los nodos (solo tenemos uno) que componen la réplica set del shard.

Ahora vamos a configurar una colección como fragmentable para que se pueda distribuir a través de los shards que tengamos

```
sh.shardCollection("mundo.ciudades", { "provincia" : "hashed" } )
```

```
[direct: mongos] test> sh.shardCollection("mundo.ciudades", {"provincia":"hashed"})
{
  collectionsharded: 'mundo.ciudades',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1708096718, i: 42 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1708096718, i: 42 })
}
[direct: mongos] test>
```

Con este comando hemos creado una base de datos llamada mundo. En esta base de datos tendremos una colección llamada ciudades. Dentro de esa colección estamos definiendo la clave shard mediante el campo "provincia".

Este campo tiene una cardinalidad limitada, 4 en el caso de Galicia, pero suficiente para nuestro ejemplo.

Aprovecharemos para insertar documentos en esa base de datos. En primer lugar cambiamos de base de datos activa a mundo y después insertamos.

```
use mundo
db.ciudades.insertMany( [{
		"nombre":"Vigo",
		"provincia":"Pontevedra",
		"coordenadas":[42.235833,-8.726667],
		"superficie":109.06,
		"población":293652,
		"autoridad": {
			"nombre": "Abel Caballero",
			"partido":"PSdeG-PSOE" },
		"hermanada": ["Victoria de Durango","Lorient","Narsaq","Oporto","Buenos Aires","Caracas","Las Palmas","Celaya","Qingdao"],
		"actualizado":ISODate("2024-01-01")
},

{
		"nombre":"Pontevedra",
		"provincia":"Pontevedra",
		"coordenadas":[42.433611, -8.6475],
		"superficie":118.22,
		"población":82535,
		"autoridad": {
			"nombre": "Miguel Anxo Fernández Lores",
			"partido":"BNG" },
		"hermanada": ["San José","Santo Domingo", "Salvador de Bahía", "Merlo", "Barcelos", "Vila Nova de Cerveira", "Gondomar",
"Naupacto"],
		"actualizado":ISODate("2024-01-01")
}
,
{
		"nombre":"Santiago de Compostela",
		"provincia":"A Coruña",
		"coordenadas":[42.883333,-8.53333],
		"superficie":220.01,
		"población":98687,
		"autoridad": {
			"nombre": "Goretti Sanmartín Rei",
			"partido":"BNG" },
		"hermanada": ["Córdoba","Santiago de Cali","Temuco","Buenos Aires","Cáceres","Qufu","Santiago de Cuba", "Pisa", "Asis",
"Coímbra","Popayán"],
		"actualizado":ISODate("2024-01-01")
}
,
{
		"nombre":"Ourense",
		"provincia":"Ourense",
		"coordenadas":[42.33556,-7.86406],
		"superficie":85.20,
		"población":104250,
		"autoridad": {
			"nombre": "Gonzalo Pérez Jácome",
			"partido":"DO" },
		"hermanada": ["Vila Real", "Tialnepantla","Quimper","Ciudad Bolivar"],
		"actualizado":ISODate("2024-01-01")
}
,
{
		"nombre":"Lugo",
		"provincia":"Lugo",
		"coordenadas":[43.011667,-7.557222],
		"superficie":329.78,
		"población":98214,
		"autoridad": {
			"nombre": "Lara Méndez López",
			"partido":"PSdeG-PSOE" },
		"hermanada": ["Ferrol", "Dinan", "Viana do Castelo","Qinhuangdao"],
		"actualizado":ISODate("2024-01-01")
}
```

```
])
```

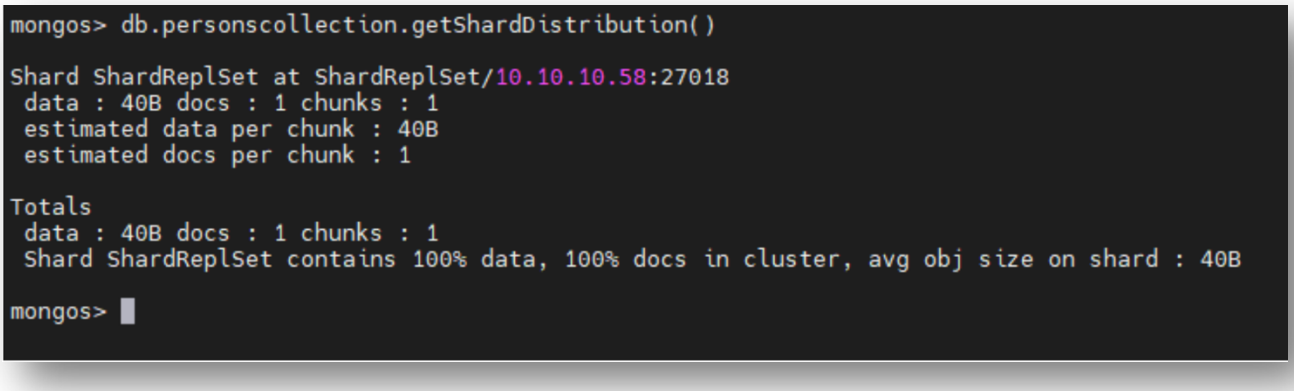
Si ahora nos conectamos al proceso mongos y hacemos un “sh.status()” podremos ver cómo están los documentos de la colección repartidos. Al tener un único nodo shard todo debe quedar guardado en él.

Estudiando la salida del comando podremos comprobar:

- Los nodos shard activos.
- Las bases de datos con colecciones fragmentadas
- Los chunks que componen dicha colección
- Los límites del dominio de la clave que tiene cada chunk y el nodo shard en el que se aloja.

MongoDB proporciona otros comandos para verificar si el sharding está trabajando según lo previsto.

```
db.ventas.getShardDistribution()
```



Con este comando podemos ver información relativa a cantidad de datos y documentos almacenados en un chunk así como la proporción relativa al conjunto del clúster.

Ventajas e inconvenientes

Como ventajas claras del sharding tenemos:

- Es más sencillo y rápido encontrar los datos que se basan en la clave de sharding puesto que ya tienen un índice previamente creado y tanto mongos en primera instancia, como el config server en el peor de los casos conocen qué nodo almacena esos datos.
- Se puede añadir más capacidad de almacenamiento y lectura sin paradas del sistema. El escalado horizontal permite añadir nodos en caliente.

- Siempre es más económico tener muchos nodos que un único pero super potente nodo.
- Es fácil combinar sharding con técnicas de alta disponibilidad como las que proporciona el Réplica Set.
- Las bases de datos orientadas a Big Data incluyen mecanismos para configurar un clúster de sharding.

Entre los inconvenientes de sharding podríamos citar:

- Un clúster sharding complica mucho la infraestructura de red.
- La latencia esperada suele ser alta.
- Consultas que requieran un JOIN entre distintos shards tienen un tiempo de ejecución alto.
- La elección de la clave de sharding puede no ser la adecuada.
- La elección de cuantos nodos shard disponer puede no ser la adecuada.

Conclusiones

Cuando una base de datos crece hasta el punto de empezar a causar lentitud en las aplicaciones cliente llega el momento de plantear una estrategia que lo solucione.

Aplicar sharding puede ser una buena opción cuando la base de datos maneje un alto volumen de información, tenga un número alto de operaciones de lectura y escritura, y cuando los datos necesiten estar disponibles.

Los shardings permiten que la base de datos se expanda en capacidad y potencia de procesamiento y, al combinarla con los conjuntos de réplica las posibilidades se multiplican. Toda esta potencia solo tiene sentido si planificamos y conocemos como se van a distribuir los datos, como van a interactuar las consultas con las colecciones y con como crecerá la base de datos.